Robot Manipulators

Constructing a High-Performance Robot from Commercially Available Parts

BY CHRISTIAN SMITH AND HENRIK I. CHRISTENSEN

A large number of robot manipulators have been designed over the last half century, and several of these have become standard platforms for R&D efforts. The most widely used is, without a doubt, the Unimate PUMA 560 series. The general availability of a platform at a reasonable price is important to allow the design of systems that can be replicated and further developed by others. Recently, there have been attempts to utilize standard platforms, as exemplified by the learning applied to ground robots (LAGRs) program organized by Defense Advanced Research Projects Agency (DARPA) [1]. The RobotCub project has also made a few robots available to the research community [2].

As actuation systems have become more powerful and miniaturized, it has become possible to build dynamical robot systems to perform dynamic tasks. Early examples of dynamic robot control include the Ping-Pong playing robot at Bell Labs [3] and the juggling robot developed by Koditschek et al. [4], [5]. Other examples of dynamic systems are walking robots [6].

However, for research work, it is often a challenge to get access to a high-performance robot, which is also available to other researchers. In many respects, robotics has lacked standard systems based upon which comparative research could be performed. Too much research is performed on a basis that cannot be replicated, reproduced, or reused. For basic manipulation, there has until recently been limited access to lightweight manipulators with good dynamics.

KUKA and DLR (a German aerospace center) have announced a new manipulator scheduled to be on the market by late 2008, but so far, the system is only marketed in Europe, and the price is expected to be high. In this article, we describe the design of a high-performance robot manipulator that is built from components off the shelf to allow easy replication. In addition, it was designed to have enough dynamics to allow ball catching, which in reality implies that the system has adequate dynamics for most tasks.

In the "Design Procedure" section, we present an application requiring significant dynamic performance and the design of a platform that fulfills the requirements. The construction of the platform is described in the "Implementation" section, and in the "Performance" section, we present our first experimental evaluation. A photo of the final implementation is shown in Figure 1.

Design Procedure

This section provides an initial analysis of the requirements for a system to perform teleoperated ball catching. A design for the system is developed from the analysis of requirements, and the performance of the design is verified by simulation.

Digital Object Identifier 10.1109/MRA.2009.934825

1070-9932/09/$26.00©2009 IEEE

A highly dynamic robot arm will pose a potential hazard to both its operator and itself, unless sufficient precautions are taken.

Experimental Requirements

The main type of experiments that we want to perform involve catching a ball thrown across a room. We anticipate a slow underhand throw from a distance of approximately 5 m. In an indoor environment, a ball can be thrown with a reasonable accuracy along a parabolic path with an apex of 2.3 m, with both the thrower and the catcher situated at a height of 1 m, as in Figure 2. Simple studies of human performance indicate that the system must be able to accommodate variations in accuracy corresponding to catching the ball within a 60×60 cm^2 window. From these requirements, we can compute flight time and velocities for the scenario, as summarized later:

- throwing distance will be approximately 5 m
- flight time will be up to 1 s, and the typical time is expected to be 0.8 s
- the ball will travel with an approximate velocity of 6 m/s at the time of arrival
- the ball should be caught if it comes within a 0.6×0.6 m^2 window.

Figure 1. *The high-performance manipulator.*

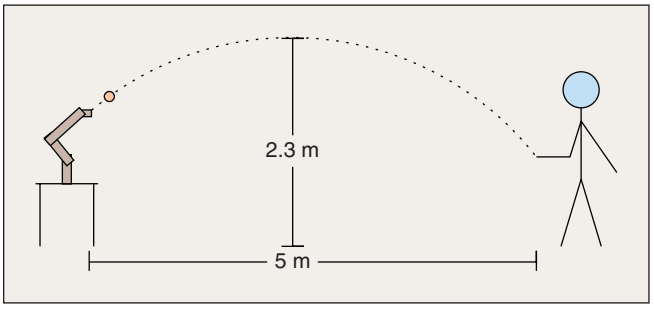

Figure 2. *Schematic of ball-catching experiment.*

Platform Requirements

One desired feature is to use standard video cameras for trajectory estimation. With 50-Hz cameras, the frame time is approximately 20 ms, and a similar time window is expected to be needed for segmentation and position estimation. In addition, at least three frames are required for trajectory estimation, but limited camera accuracy will mean that more possibly, as many as ten images might be necessary [7]. Thus, the time delay for the initiation of a throw to the initial trajectory estimate might be 200 ms. We also intend to do teleoperated catching, where a human operator controls the robot, so we have to allow for the operator's reaction time. This might be around 100 ms, so 300 ms is reserved for the initial reaction to a throw, leaving 500 ms for the arm to move into position. In the worst-case scenario, the arm has to move against gravity from one opposing corner of the operational window to another, a distance of 0.9 m. Since the initial experiments will not be concerned with grasping, a passive bucket-type end effector will be employed, and the positioning error must be smaller than the radius of the bucket, preferably less than 1 cm. These requirements can be summarized as follows:

- end effector has to move 0.9 m in 0.5 s, (partially) against gravity, from stand-still to stand-still
- the precision of positioning the end effector should be within 1 cm.

Given constant acceleration and deceleration, 0.9 m can be traveled in 0.5 s if the acceleration is at least 14.4 m/s^2, and the maximum velocity is at least 3.6 m/s. These are the minimum requirements: the actual implementation should have some margin to allow for uncertainties. This requires the control to be performed in real time, so it is desirable to have closed-form solutions for kinematics that can be calculated fast. This puts constraints on the overall kinematic structure.

To enable flexibility in the design of future experiments, it should be possible to mount different types of sensors in the end-effector reference frame, so this should have six degrees of freedom (6 DoF) and be freely orientable in a dexterous manner.

A highly dynamic robot arm will pose a potential hazard to both its operator and itself, unless sufficient precautions are taken. The control of the arm must be sufficiently precise so that safe paths can be accurately followed, and precautions against malfunctions must be taken. The former requires high-frequency/low-latency control loops, and the latter that software and hardware malfunctions are kept at a minimum and that the negative effects of malfunctions are minimized. Thus, the software environment should be a stable real-time system, and the hardware contain fail-safe fallback for dealing with software failure. These requirements are summarized as follows:

- closed-form analytical kinematics and dynamics
- at least 6 DoF
- acceleration of at least 14.4 rm/s^2 for end effector
- velocity of end effector of at least 3.6 m/s
- a stable real-time system and fault-tolerant hardware.

Designed Solution

As detailed in "Notes on Manipulators," there are a number of fairly fast robotic manipulators commercially available, like for

Notes on Manipulators

Cost Breakdown

The total cost of hardware used in the setup was just below €50,000. For a detailed cost breakdown, see Table S1. Please note that these are the actual prices paid and that there is no guarantee for future availability at these same prices.

Comparison to Alternatives

Table S2 shows a summary of alternative manipulators in more or less the same performance and/or price segment. Performance figures are taken from manuals provided by the manufacturers. Prices are either quotes or actual-paid prices. It should be noted that pricing may vary significantly:

- actual prices paid or as quoted by supplier
- rated power consumption
- used price varies with condition
- available in different configurations
- not tested for durability.

The proposed robot compares well to other options. The KR5 may have a more attractive performance/price ratio, but the proprietary interface limits control to high-level position or velocity control at 80 Hz with no low-level interface, so it may not be suitable for some research applications. In contrast, the KUKA lightweight robot (LBR) has an accessible interface and torque sensors in all joints [8] but is substantially more expensive.

Different manufacturers provide different performance metrics, especially for velocity which is given in either joint or Cartesian space, making comparison less straightforward. Power-to-mass ratios may give an indication of performance, but it is worth to bear in mind that power also correlates inversely with safety and that less-powerful models may often be better suited for operation in close human proximity.

Other Applications

Apart from the automated ball-catching task described in the ''Experimental Setup'' subsection, the manipulator has also been successfully applied to other tasks, such as teleoperated ball catching, robot control using a Wiimote video-game controller, and positioning visual targets used for automated camera calibration. Since the platform was designed for a task requiring high velocities, it has adequate performance for tasks that require lower velocities as well. The strength of the setup has shown to be the ease with which it can be applied to new tasks, given a completely open interface. An obvious weakness is the high power consumption, making it unsuitable for mobile applications in spite of the relatively low weight. Planned future applications include adding force and torque sensors to enable teleoperated force control.

Project Web Site

As part of the effort to increase the availability of the proposed platform, there is a Web site with information regarding the platform as well as downloadable media and source code at www.cas.kth.se/∼ccs/Robot_arm.

Table S1. Prices (in euro) for the setup used in the present article.

Part(s)	Price (€)
Actuators	38,000
Rigid links	3,400
CAN system	1,600
Mountings	500
Power supply	4,400
Control computer	1,600
Total	49,500

Table S2. Comparison to some alternative manipulators.

Name	Price[a] (€)	DoF	Reach	Weight (kg)	Power[b] (Peak)	Payload (kg)	API	Velocity
PUMA560	—[c]	6	0.86 m	63	1.5 kW	2.5	RT joint	0.5 m/s
Neuronics Katana	20,000	5	60 cm	4.3	96 W	0.4	RT traj/joint	90°/s
KUKA KR5 850	22,000	6	0.85 m	29	2.3 kW	5	80 Hz pos/vel	250°/s
Schunk LWA3	45,000	7	—[d]	≈10[d]	0.48 kW	5	joint traj/current	70°/s
Proposed manipulator	50,000	6	91 cm	23	5.5 kW	—[e]	600 Hz position/velocity	7 m/s
Barret WAM	70,000	7	1 m	27	250 W	3	500 Hz traj/force	1 m/s
KUKA LBR	120,000	6	0.94 m	14	720 W	14	1 kHz torque/force/traj	120°/s

Data as given in manufacturers' documentation.
[a]Actual prices paid or as quoted by supplier.
[b]Rated power consumption.
[c]Used price varies with condition.
[d]Available in different configurations.
[e]Not tested for durability.

Table 1. Specifications for the parts used in the manipulator.

Part	Product Name	Mass (kg)	Comment
First joint	PowerCube PR110	5.6	51:1 reduction gear
First link	PAM104	0.2	55 mm cylindrical rigid link
Second joint	PowerCube PR110	5.6	101:1 reduction gear
Second link	PAM108	0.8	200-mm cylindrical rigid link
Third joint	PowerCube PR110	5.6	51:1 reduction gear
Third link	PAM119	0.2	45-mm conical rigid link
Fourth joint	PowerCube PR070	1.7	51:1 reduction gear
Fourth link	PAM106	0.6	200-mm cylindrical rigid link
Fifth, Sixth joint	PowerCube PW070	1.8	2-DoF wrist joint

instance, the KUKA lightweight arm [8]. It has been shown to be fast enough to catch thrown balls autonomously [7], but needs a very early ballistic path estimate to do this. In our experiments, we also want to include a human operator in the control loop to do teleoperated catching, so we require even faster movements to compensate for slow human reactions. With perhaps only half the time to get into position, twice the speed is needed.

To cater to the special needs of our experiments, we decided to construct our own 6-DoF arm to examine if this could be done using PowerCube modules from Amtec. These modules are available off the shelf and allow rapid prototyping. The range of modules clearly includes some that have specifications adequate for the target application (see the "Hardware Implementation" subsection). These modules also have a built-in controller that can be used for embedded safety functions.

The actual performance depends on the configuration that the modules are assembled in, so a few different configurations were examined more closely in computer simulation, where a 10% uncertainty was added to the maker specifications. The configuration that showed the most promising one is kinematically similar to a Puma560 arm (and many other commercially available robots). Not only does this configuration allow for very good dynamic performance (see the "Simulated Performance" subsection), but as it has been widely used and studied, several implementation issues are already solved, making the design process considerably faster. For example, the closed-form solutions for inverse kinematics and dynamics are well known. Fast dynamics is achieved by keeping the moment of inertia as low as possible in the moving parts and placing heavier, more powerful modules where their impact on the inertial load is lower. In the final design, three 1.5-kW motors are used to position moving parts, weighing approximately 10 kg. Also, the arm is designed so that the center of mass will be close to the rotational axis of the first joint when working in the intended window of operation. This will balance the arm and keep down the strains on the first joint.

The choice of gear ratios and link lengths induces a tradeoff between acceleration and maximum velocities. This was balanced to minimize the time needed to move the end effector from one stationary position to another within the operation window. Since there is a limited, discrete amount of possible combinations of actuators, the optimum could be found with an exhaustive search. The resulting configuration with the best simulation performance is specified in Table 1. The design and dimensions can be seen in Figure 3. The workspace is more than large enough to accommodate for the specified 60×60 cm^2 window for ball catching. A cross-section of the workspace can be seen in Figure 3(c). The arm has rotational symmetry as viewed from above, but motion is limited to a half circle to avoid collisions with any objects behind it.

The control setup should have as short loop times as possible. The PowerCube modules support several different communication options, but for robustness and responsiveness, 1 Mb/s controller area network (CAN) bus was deemed optimal. This can be implemented in several different ways. In principle, all modules could be on a single CAN bus, or each module could have a bus of its own. The last joint is a combined pan-tilt unit, which uses a single bus to control both degrees of freedom. Depending on the number of modules per bus, the lengths of the control cycle will vary (see the "Control Loop Times" subsection). This means that the control computer could be equipped with either one, two, or three CAN controllers for symmetric loads, or four or five controllers for asymmetric loads, where the inner joints that control positioning are run at a higher frequency than the outermost controlling orientation. Simulations where the inner joints were controlled at 500 Hz and the outer joints at 200 Hz show that this is a viable option. In simulation, the inner joints can be stably controlled at full

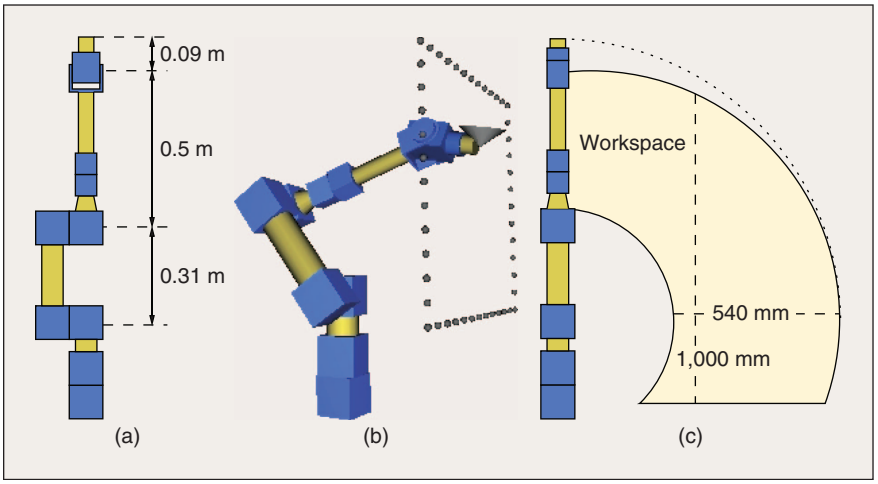

Figure 3. *The first design of the manipulator using Amtec PowerCubes. (a) Dimensions. (b) Three-dimensional (3-D) rendering of arm and operational window. (c) Workspace with the tool oriented toward user.*

power output using frequencies from 400 Hz and upward, but the real implementation may have slightly different requirements.

The first choice for the computer performing the direct low-level control of the robot was real-time application interface (RTAI), a real-time Linux system that has showed good performance in previous studies [9], but some testing led to the choice of regular Linux patched with high-resolution timers (http://www.tglx.de/hrtimers.html), as this not only showed better real-time performance but also allowed easier implementations in user space. The control computer will also perform the trajectory generation and dynamic and kinematic calculations.

Teleoperation should be enabled by connecting to an external computer for the user interface (UI). The communication with the UI computer should be in Cartesian space, since the kinematic structure of the arm allows for eight different joint-space configurations for each Cartesian position, and the choice of configuration should be made locally by the low-level controller for best performance. The connection is made over user datagram protocol (UDP)/IP, as this can give significantly better control performance than transmission control protocol (TCP)/IP over Internet connections [10]. The connection to the UI will not need hard real-time performance, but the smaller the time lag can be made, the better the performance. In early experiments over our LAN, the round-trip time from the UI input via the manipulator controller to UI feedback has been shown to be 10–20 ms. A schematic of the connection architecture is shown in Figure 4. The basic specifications of the designed solution are as follows:

- 6-DoF arm made with Amtec PowerCubes
- kinematic configuration of Puma560 type
- GNU/Linux, preferably with high-resolution timers, for control computer
- communication over several parallel CAN connections.

Simulated Performance

The performance of the proposed arm was first calculated using the maker's specifications and numerical simulations. The results for the travel times depend on the type of controller used, and in this case, the controller included a dynamic model for feedforward control, and the torques in each individual joint were set to achieve a target velocity as quickly as possible. The target velocity was chosen as the minimum of the actuators' maximum velocity and the highest velocity from which stopping at the desired position was achievable. This latter factor was calculated using the maximum torque of the actuators and the inertial load of the current configuration, a figure that was then decreased slightly to achieve a margin. Using this simple controller, the simulated arm had more than adequate performance, as is summarized in Table 2. The first acceleration figure given is the maximum achievable for small movements, and the second figure for larger, cross-workspace movements.

Implementation

This section describes the technical details of the actual implementation of the robot arm.

Fast dynamics is achieved by keeping the moment of inertia as low as possible in the moving parts and placing heavier, more powerful modules where their impact on the inertial load is lower.

Hardware Implementation

The arm proposed and specified in the earlier sections was constructed and mounted on a sturdy industrial work table (see Figure 1). The lower three actuators have a maximum output of almost 1.5 kW each, harmonic drive gearboxes, and incorporated brakes to lessen motor strain when not moving. The fourth actuator is similar, but considerably smaller, as it carries a lighter inertial load. The maximum output is 0.36 kW. The last two joints are contained in a combined pan/tilt unit. This is less powerful, but has lower weight per joint than other solutions. This incorporates the same gearbox and

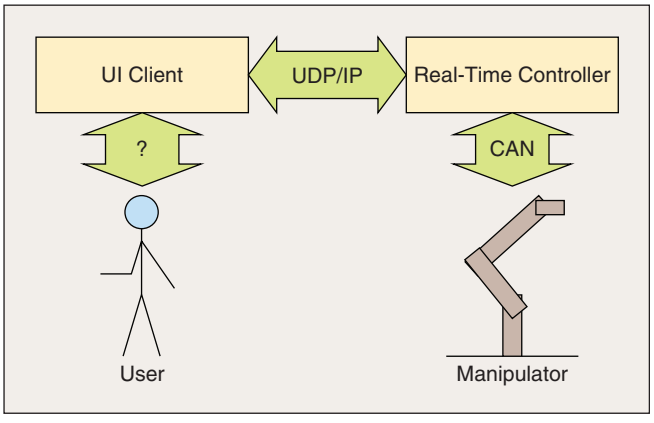

Figure 4. *Schematic of the connection architecture for a teleoperation scenario.*

Table 2. Simulated performance of robot arm.

Endpoint acceleration	>100 m/s^2 (>30 m/s^2)
Endpoint velocity	>5 m/s
Travel time across window vertically, from standstill to standstill	<0.36 s
Travel time across window diagonal, from standstill to standstill	<0.37 s
Travel time from window center to upper corner, from standstill to standstill	<0.22 s
Repeatability of position	±1 mm

Performance is dependent of arm position, so all values are given as their lower limit within the window. The first acceleration given is the maximum for small movements, and the second (in braces) is the maximum for large movements.

brakes as the other modules. Specifications can be found in Tables 1, 3, and 4.

The PowerCube modules have a simple onboard controller with basic security features. They will not allow motion beyond user-settable angle limits and will perform an emergency stop if these limits are exceeded or if no watchdog signal has been transmitted for 50 ms. The joint angle limits are set to avoid collisions with self or environment (Table 5). There are two sets of limits, each set prohibiting collisions in itself but with a limited workspace. The system will switch limit sets when moving out of range of one set and into range of another, with an intermediate limit set that consists of the tighter limits of the two sets. This limits each individual module to a safe interval, even if communication were to break down halfway through a limit switch, while at the same time, allowing the robot to use a large part of the potential workspace.

Two tests of the safety measures were carried out. First, the communication link was severed between the computer and the robot. This results in a termination of the watchdog update, and the modules finish their last command and engage the brakes. In the second test, illegal position commands were intently issued by the control program. The modules' onboard controller correctly identified these as violating joint limits. The arm moved into the legal position closest to the commanded position and stopped. This accounts for safe handling of an unexpected breakdown of control algorithms, the control computer, or the CAN communication link.

A power supply unit capable of delivering the required 30 A at 48 V to each module was constructed with Cosel PBA-1500 F power converters. An emergency stop that works by directly cutting the power was implemented so that the power unit cannot run without the being emergency stop present. The emergency stop has been verified to stop the modules and engage the brakes.

The communication interface was designed to be implemented over four separate CAN buses: one each for the three inner (position controlling) joints and one common bus for the three outer (orientation controlling) joints. Two, two-channel peripheral component interconnect (PCI) CAN controllers from Kvaser were chosen, as these had open-source Linux drivers that seemed plausible to port to real-time usage. A 3.6-GHz Pentium 4 Dell PowerEdge 1800 server was acquired to use as control unit, since it provides a good balance of processing power and reliability.

Table 3. Manufacturer's specifications for the joint actuators.

Joint No.	Max. Torque (N · m)	Max. Angular Velocity (rad/s)	Repeatability (rad)
1	134	8.2 (470°/s)	± 0.00035
2	267	4.1 (238°/s)	± 0.00035
3	134	8.2 (470°/s)	± 0.00035
4	23	8.2 (470°/s)	± 0.00035
5	35	4.3 (248°/s)	± 0.00035
6	8	6.2 (356°/s)	± 0.00035

Table 4. The Denavit-Hartenberg parameters for the arm, using J.J. Craig's notation.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0°	0 m	0 m	θ_1
2	−90°	0 m	0 m	θ_2
3	0°	0.31 m	0 m	θ_3
4	−90°	0 m	0.51 m	θ_4
5	−90°	0 m	0 m	θ_5
6	90°	0 m	0 m	θ_6

Table 5. Limits on joint angles.

Joint No.	Set 1	Set 2
1	−90° to +90°	−90° to +90°
2	−100° to −40°	−130° to −70°
3	−60° to 50°	−40° to 90°
4	−160° to +160°	−160° to +160°
5	−120° to +120°	−120° to +120°
6	−180° to +180°	−180° to +180°

Software Implementation

A Linux 2.6.19 kernel was patched with high-resolution timers for low-latency real-time performance. A customized communications application programming interface (API) was implemented to guarantee low-latency communication with the modules as well as customized vector manipulation libraries optimized for calculating arm dynamics. The control loop is run in soft real time. Tests have shown that the worst-case latency of this setup is less than 100 μs, which is sufficient. The average jitter for the main loop of the control algorithm is 6 μs, which is significantly less than the modules' latency of up to 600 μs. This soft real-time performance is comparable to that of hard real-time systems like RTAI, but with the advantage of simple straightforward user-space implementation.

Inverse kinematics and dynamics are calculated using the analytical solution for a Puma arm in [11], and the forward dynamics is calculated using the second algorithm in [12]. Inverse kinematics can be calculated in 1.7 μs, and dynamics in 41 μs, so that all calculations needed in the control loop take less than 50 μs. This means that, virtually, all latency in the control loop originates from the communication with the modules over the CAN bus.

Combined position and velocity control is implemented on the system using a combined feedforward-computed torque control (CTC) scheme and a feedback proportional integral (PI) controller. When a new set point enters the controller, a velocity ramp trajectory is calculated in joint space. This trajectory is limited by a preset top velocity (presently, 4 rad/s) and a maximum acceleration, but is otherwise the shortest path to the desired position θ_d and velocity $\dot{\theta}_d$ from the actual position θ_a, without exceeding the maximum allowable acceleration a_{max}, see (1). This ramp works under the assumption that

the desired position is frequently updated to new positions in accordance with the desired velocity.

$$\dot{\theta}_{\text{ramp}} = \sqrt{|\theta_{\text{d}} - \theta_{\text{a}}| \cdot a_{\max}} \cdot \text{sign}(\theta_{\text{d}} - \theta_{\text{a}}) + \dot{\theta}_{\text{d}}. \quad (1)$$

The maximum acceleration $a_{\max}$ is limited by a preset limit value and the maximum achievable acceleration, computed by calculating the acceleration produced by maximum torque and taking away a small safety margin. (The limits on velocity, acceleration, and jerk are chosen to limit the mechanical stress on the system, while still being able to reach a given point in the workspace in less than 0.5 s.) The ramp is recalculated in each iteration of the control loop using the current position and velocity. The desired acceleration fed to the CTC controller is the one necessary to achieve the target velocity $\dot{\theta}_{\text{ramp}}$ as soon as possible, without violating the limits on acceleration or jerk. The CTC controller then uses the inverse dynamics function to determine the necessary torques to follow the trajectory. These torques are converted to currents and sent to the actuator modules. For evaluation purposes, the acceleration has been limited to 16 rad/s^2 and jerk to 400 rad/s^3.

A PI controller monitors the difference between desired velocity and actual velocity and corrects the controller current accordingly. This corrective term is necessary, as the feedforward CTC controller does not contain an accurate-enough model of friction, the movements of power cords, or the nonlinearities in the current/torque relationship. For a schematic of the control scheme, see Figure 5.

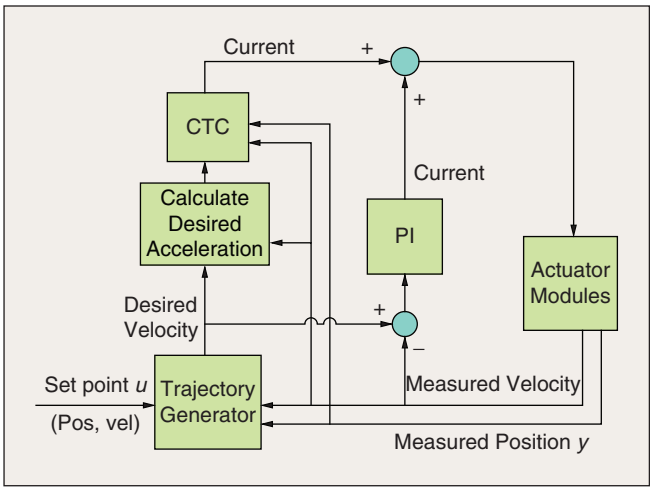

Figure 5. *Schematic of controller.*

Performance

There is still some fine-tuning remaining to be done for the robot arm, but even so, it already fulfills all the specified requirements and has a performance similar to the simulation.

Precision

The repeatability of positioning was measured by fixing a paper target with a millimeter scale to the last joint of the arm. The arm was stopped in the center of the workspace. A laser pointer producing a light point 1 mm in diameter was fixed to point at the center of the target. The arm was then moved around a complicated path traversing and circling the workspace of approximately 1 min. The arm was then returned to the original position. To the precision of the scale and the observer's perception, the pointer was in the middle of the target. This was repeated for different positions and angles, with the laser pointer mounted both horizontally and vertically, with the same results. The repeatability is therefore at least ± 1 mm. The arm has also been tested to follow a straight path with submillimeter accuracy, but this has only been performed at very low speeds for safety reasons, so there are no figures for the accuracy at higher velocities.

Dynamic Performance

The arm has been timed to traverse the operational window shown in Figure 3(b) vertically (distance 60 cm) in both directions in 0.39 s from standstill to standstill, as predicted in the simulations. As for other movements, horizontal (60 cm) and diagonal (90 cm) traversion, only times of 0.5 s have been verified, as this is enough for our application as we want to minimize mechanical stress on the equipment. However, this implies that any point-to-point motion in the operational window takes at most 0.5 s to execute. The outermost joints are slightly slower than the inner ones, so the final angular alignment of the end effector rather than the positioning is the limiting factor for many configurations.

Control Loop Times

The modules are specified to handle CAN bus communication up to 1 Mb/s, but experiments show that this rate cannot be maintained continuously. Especially when controlling several modules on a single CAN bus, there is a tendency for CPU overload/overheat in the modules. This results in an error that requires a shutdown and cooldown before the operation can be resumed. The communication frequencies anticipated from the specifications can be seen in Table 6. The time to complete a communication loop consists of 0.134 ms needed to send a

Table 6. Theoretical control loop speeds over the CAN bus.

	Modules Per CAN Controller Card			
	1	2	3	6
Cycle periods at 1 Mb/s				
With velocity polling (ms)	1.04	1.30	1.87	3.22
Without velocity polling (ms)	0.52	0.65	0.8	1.61
Cycle periods at 500 kb/s				
With velocity polling (ms)	1.57	2.14	3.22	6.43
Without velocity polling (ms)	0.79	1.07	1.61	3.22
Control frequency at 1 Mb/s				
With velocity polling (Hz)	961	769	535	311
Without velocity polling (Hz)	1,923	1,538	1,250	621
Control frequency at 500 kb/s				
With velocity polling (Hz)	637	467	311	156
Without velocity polling (Hz)	1,265	935	621	311

This soft real-time performance is comparable to that of hard real-time systems like RTAI, but with the advantage of a simple straightforward user-space implementation.

CAN message at 1 Mb/s (or 0.268 ms at 500 kb/s), and at approximately 0.25 ms, a module needs to respond to a request. The response time varies with the type of request. When performing several read/writes to different modules on the same bus, part of the time spent waiting for one module's response can be used to communicate with another, hence, the slight nonlinearity in loop times as a function of the number of modules. The table shows two different speeds for each setup, with or without velocity polling. The modules have internal velocity measurements that are more accurate than just differentiating two position measurements. However, if these velocity measurements are used, the temporal resolution will be lower because of the extra time needed for sending this additional data. Experiments have yet to show which strategy will yield the best overall performance.

In the implementation, a control loop frequency of 600 Hz for the inner three cubes and 200 Hz for the outer three cubes is used. This is with velocity polling, at 1 Mb/s, using one CAN bus per card for the inner cubes and a joint bus for the outer ones. The lower frequency is obtained by only communicating with one of the outer cubes in each iteration of the control loop. Because of their limited inertia and power, the outer cubes have a limited influence on the overall dynamic performance of the arm, and the error induced by scarce measurements from the outer cubes is negligible. The communication frequency is 37.5% lower than the theoretical maximum, but at this frequency, the control loop can run uninterrupted for hours without overheat. Since the lower-than-specified frequency is accomplished by padding the loop, the padding also absorbs the variations in module response time, resulting in virtually no variations in loop-cycle times.

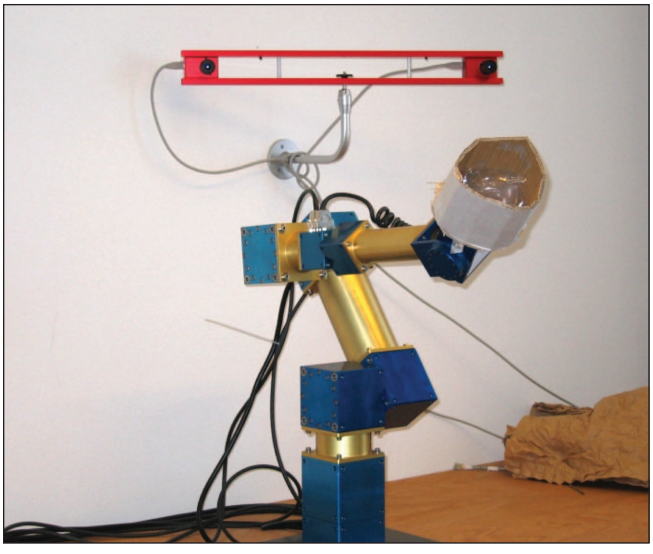

Figure 6. *The manipulator with cameras and ball-catching end effector.*

Calibration Loss

When performing ball-catching experiments, the robot's movements are fast, but centered in the designated workspace [the square operational window shown in Figure 3(b)], and the typical duration is not very long. In teleoperation experiments, we have repeatedly let users move the manipulator freely for up to 30 min. In these longer, free sessions, we have noticed a tendency for the joint encoders to lose calibration. The faster the movement is and the farther it is from the designated workspace, the larger is this tendency.

Some simple tests were performed to verify this behavior. When only moving within the designated workspace, there was no measurable loss of calibration, even for movements at full capacity. For motions far outside the workspace, there was no measurable loss when moving at angular accelerations below 3 rad/s^2. With higher acceleration settings when moving outside the work space, loss of up to a few degrees of calibration has been observed in the three lower modules, especially for irregular motions. Recalibration of the encoders is easily done by returning the manipulator to a predefined home position, but this disrupts whatever other motion was being performed.

Ball-Catching Experiments

To verify the performance of the manipulator, a setup allowing for an autonomous ball-catching scenario was constructed. These experiments are still at an early stage, but the early results are promising.

Control Server

A first prototype server application has been implemented. It receives position and velocity set points in Cartesian coordinates from a client computer over an UDP/IP connection and returns information on present position and velocity in both Cartesian and joint space. All commands and measurements are timestamped to enable correction for time lags over the communication link.

Experimental Setup

The manipulator was fitted with an end effector consisting of a passively damped cylinder with a 14 cm diameter (see Figure 6). We launched soft juggling balls from a distance of approximately 4 m. To ensure repeatability, the balls were launched from a mechanical launcher with a precision of ± 10 cm for this distance. The balls have to hit within 4 cm of the center of the cylinder to be caught without bouncing off.

Using this setup, the flight time of the ball was approximately 0.8 s. The ball position was measured with stereo cameras mounted on a 0.6 m baseline, approximately 0.5 m behind and above the robot (see Figure 6). The ball was tracked with an extended Kalman filter (EKF), as described in [7].

The ball is detected in each image using simple color segmentation. First, the 24-b red-green-blue (RGB) camera

images are converted to 24-b hue saturation value (HSV). The balls have a hue value of three, and a (largely varying) saturation value of approximately 160, so all pixels in the range 1–5 for hue and 120–200 for saturation are preliminarily marked as ball pixels. A second pass that only keeps marked pixels with at least three marked neighbors eliminates noise. The center of mass of the marked pixels is calculated and used as the ball centroid. To decrease the segmentation time, a subwindowing scheme similar to the one proposed in [13] was used. After the ball has been detected the first time, only a subwindow where the ball should be expected to be found is processed. This subwindow is calculated using the state estimate from the EKF, and the size of the window is set to cover several times the standard deviation in position. With this approach, the ball can be segmented and localized with a reasonable accuracy at less than 4 ms processing time per stereo image pair, giving sufficient real-time performance.

The catch position is decided by finding the point where the predicted ball trajectory intersects the plane of the robot's workspace. This position is then sent to the control computer, which moves the manipulator to the position. Launching 108 balls that hit within the operating window, with an average distance of 24 cm from the manipulator's starting position, 73% were caught, 19% bounced off the rim of the end effector, and 8% were missed. The main cause of missed catches was errors in the early predictions of the ball path, causing the robot to start moving in the wrong direction.

Conclusions

In this article, we have presented the requirement for a highly dynamic robotic system to be used in studies for ball catching. From these requirements and a number of secondary goals, a system has been designed using off-the-shelf actuation modules. Associated software for real-time control has been designed and implemented on a commercially available computer platform. The system operates at 600 Hz and satisfies all the requirements specified for the design. Results from early experiments demonstrate that the system fulfills the static and dynamic requirements to allow ball catching.

Acknowledgment

The research was funded in part by the sixth EU Framework Program, FP6-IST-001917, project name Neurobotics.

Keywords

Ball catching, robot design, teleoperation.

References

[1] L. D. Jackel, E. Krotkov, M. Perschbacher, J. Pippine, and C. Sullivan, "The DARPA LAGR program: Goals, challenges, methodology, and phase I results," *J. Field Robot.*, vol. 23, no. 11–12, pp. 945–973, 2006.

[2] G. Sandini, G. Metta, and D. Vernon, "The icub cognitive humanoid robot: An open system research platform for enactive cognition," in *Artificial Intelligence 50 years*, J. G. Carbonell and J. Siekmann, Eds. Berlin: Springer Verlag, 2008, pp. 358–369.

[3] R. Andersson, "Understanding and applying a robot ping-pong player's expertise," in *Proc. 1989 IEEE Int. Conf. Robotics and Automation*, Scottsdale, AZ, 1989, pp. 1284–1289.

[4] M. Buhler and D. E. Koditschek, "From stable to chaotic juggling: Theory, simulation, and experiments," in *Proc. 1990 IEEE Int. Conf. Robotics and Automation*, Cincinatti, OH, 1990, pp. 1976–1981.

[5] A. A. Rizzi and D. E. Koditschek, "Progress in spatial robot juggling," in *Proc. 1992 IEEE Int. Conf. Robotics and Automation*, Nice, France, 1992, pp. 775–780.

[6] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA: MIT Press, 1986.

[7] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2001, pp. 1623–1629.

[8] G. Hirzinger, N. Sporer, A. Albu-Schafer, M. Haahnle, and A. Pascucci, "DLR's torque-controlled light weight robot iii—Are we reaching the technological limits now?" in *Proc. Int. Conf. Robotics and Automation*, 2002, pp. 1710–1716.

[9] D. Aarno, "Autonomous path planning and real-time control—A solution to the narrow passage problem for path planners and evaluation of real-time Linux derivatives for use in robotic control," Master's thesis, Dept. Numerical Analysis and Computer Science (NADA), KTH, Sweden, 2004, TRITA-NA-E04006.

[10] S. Munir and W. J. Book, "Internet-based teleoperation using wave variables with prediction," *IEEE/ASME Trans. Mechatron.*, vol. 7, no. 2, pp. 124–133, June 2002.

[11] J. Craig, *Introduction to Robotics: Mechanics and Control*, Reading, MA: Addison-Wesley, 1986.

[12] M. Walker and D. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Trans. ASME J. Dyn. Syst. Meas. Control*, vol. 104, no. 3, pp. 205–211, 1982.

[13] I. Ishii and M. Ishikawa, "Self windowing for high-speed vision," *Syst. Comput. Jpn.*, vol. 32, no. 10, pp. 51–58, 2001.

Christian Smith's work includes reinforcement learning for social robotics at Advanced Telecommunications Research Institute International (ATR), Japan, in 2004. He received his M.Sc. degree in engineering physics from Royal Institute of Technology, Stockholm, Sweden, in 2005. He is currently a Ph.D. student at the Centre for Autonomous Systems, Royal Institute of Technology. His current research focuses on the design and control of highly dynamic teleoperated robotic systems inspired by studies of human neurophysiology.

Henrik I. Christensen received M.Sc. and Ph.D. degrees from Aalborg University, Denmark, in 1987 and 1990, respectively. He is the KUKA chair of robotics at Georgia Institute of Technology and director of Center for Robotics and Intelligent Machines. He has served as the founding director of the Center for Autonomous Systems at the Royal Institute of Technology and as a lecturer at Aalborg University. His main research is on systems integration and data fusion. He has published more than 250 contributions across vision, robotics, and artificial intelligence (AI). He has served/is serving on a large number of editorial boards. In addition, he participates in a large number of research projects across three continents. He was the founding chair of the European Robotics Network (EURON). He is a Senior Member of the IEEE, member of American Association for AI (AAAI), and an officer of International Foundation of Robotics Research (IFRR).

Address for Correspondence: Christian Smith, Royal Institute of Technology, Teknikringen 14, 100 44 Stockholm, Sweden. E-mail: ccs@kth.se.